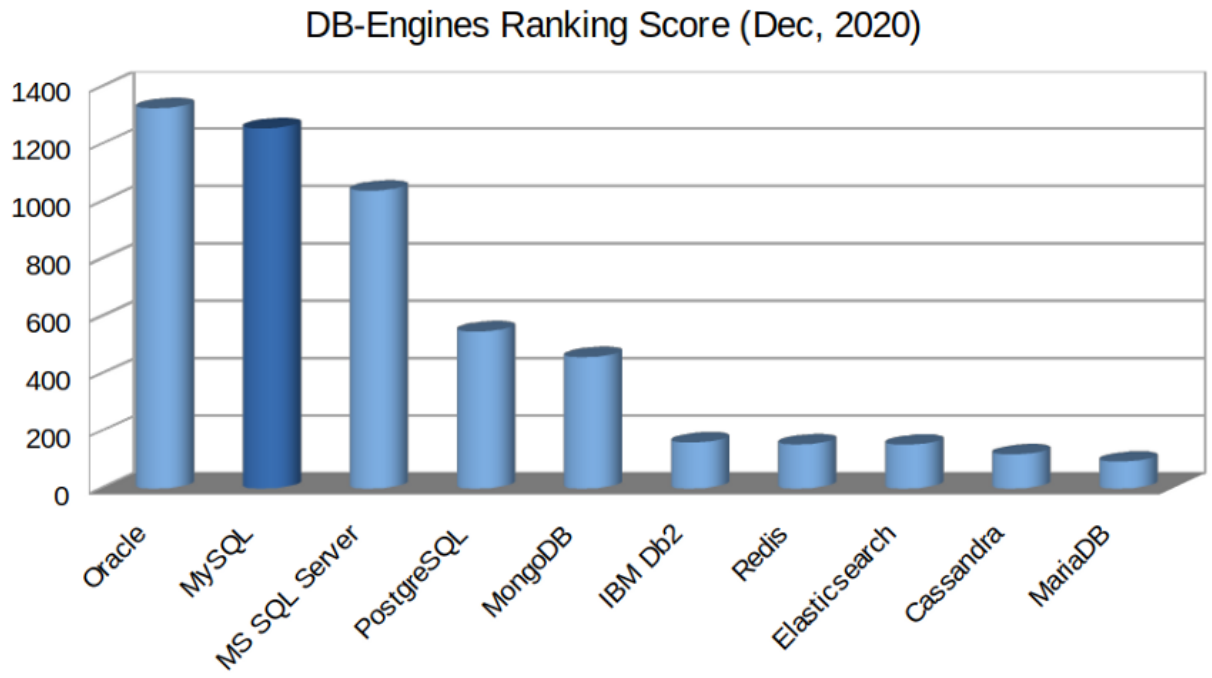
Oracle Day 1

Databases are the cornerstone of any Software Applications. You will need one or more databases to develop almost all kind of Software Applications: Web, Enterprise, Embedded Systems, Real-Time Systems, AI, ML, HPC, Blockchain, IoT, and many other applications.

With the rise of Microservices, Cloud, Distributed Applications, Global Scaling, Semi-Structured Data, Big Data, Fast Data, Low Latency Data: the traditional **SQL** databases are now joined by various **NoSQL. NewSQL,** and **Cloud** databases.

There are a whopping **343** databases at present. Here I will list popular databases from them



Different databases in the market:

Oracle

MS SQL Server

Teradata

IBM DB2

Sybase

MySQL

PostgreSQL

Natezza

2. Oracle

When **Edgar F. Codd**'s published his revolutionary paper "**A Relational Model of Data for Large Shared Data Banks**" (1970) on the Relational Database Management System (RDBMS), it has completely changed the landscape of database Systems. The paper particularly inspired a young Software Engineer **Larry Ellison** (current CTO of Oracle Corporation). He later created the world's first commercially available RDBMS system **Oracle** in 1979. Since then, Oracle remained the leading commercial RDMBS System and dominated the Unix and Linux Systems. Over the last 41 years, Oracle has evolved with time and contributed to the RDBMS and the overall database Systems innovations.

Currently, Oracle is the number one commercially supported database and one of the widely used RDBMS overall. Its latest release (21.c) has added many innovative features that will make it an attractive option in the coming years.

5 Key Features

- Proprietory RDBMS.
- Offers ACID transactional guarantee. In terms of CAP, it offers immediate Consistency as a single Server.

- Advanced Multi-Model databases supporting Structured Data (SQL), Semi-Structured Data(JSON, XML), Spatial Data, and RDF Store. Offers multiple access pattern depending on the specific Data Model
- Offers Blockchain Tables.
- Supports both OLTP and OLAP workload.

When to Use Oracle

- If a company wants to have a Converged database or Master Database (One database for OLTP and OLAP).
- Traditional transactional workloads with structured (SQL) data, and when ACID transaction guarantee is a key criterion.
- Blockchain Table is required.
- For Data Warehousing.
- A multi-model database including Geospatial Data type is an essential requirement.

When not to Use Oracle

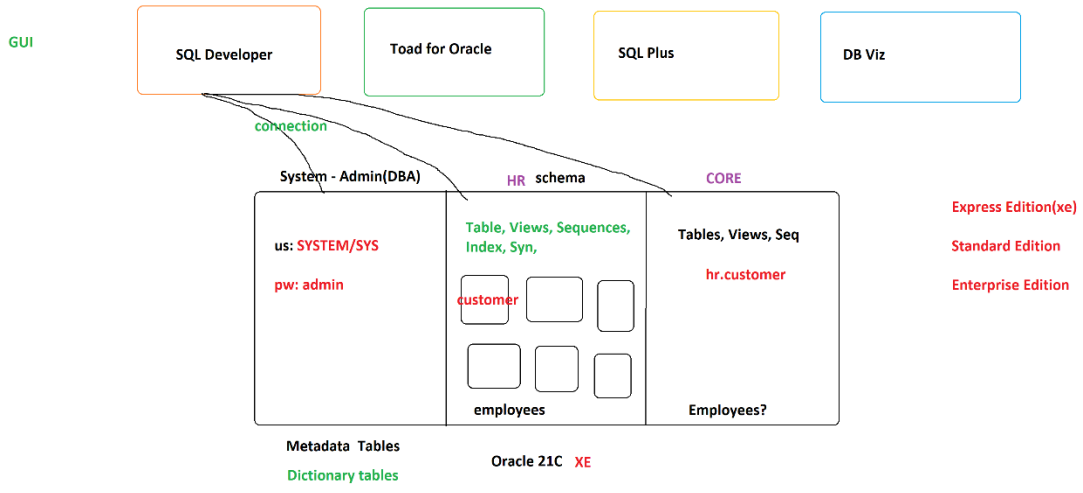
- If a company wants to save money on a database.
- Multi-Master ACID transaction is a must-have feature.
- Data is Semi-structured, i.e., JSON data with advanced query functions.
- Data is extremely relational (e.g., Social Media), i.e., Graph like data.

Oracle As a Service

- Oracle Converged Database
- Amazon RDS for Oracle

In the past, almost all databases were relational. They used a set data structure, which allowed them to link information from different “tables”, using indexes. These data “buckets” could then be linked through a “relationship”. SQL (Structured Query Language) is the language used for this kind of databases. It provides commands to create, retrieve, update, and delete information stored in the tables.
NoSQL, then, stands for “No Structured Query Language”. It is a non-relational type of database. In this case, databases do not use any kind of relational enforcement. The architect of the database determines what relationships, if any, are necessary for their data, and creates them.

Oracle Set up:



SQL -- Structured Query Language

ANSI

Data Types in Oracle

1.Numeric

int

decimal

float

double

number

number(6) 999999

number(8,2) 999999.99

number(2,2) 0.99

2. Character

char 2000

varchar 2000

varchar2 4000

first_name char(10) ARUN + 6 char (reserved)

first_name varchar2(10) ARUN + 6 char released

3. Date

date

insert -- format mm/dd/yyyy
dd/mm/yyyy

4. LOB

CLOB -- GB

```
create table test
(
cust_id number(2,2),
cust_name char(10),
cust_name2 varchar(10)
);
```

```
drop table test;

insert into test values(0.89,'Arun','john');

desc test;

select * from test;

select length(cust_name),length(cust_name2) from test;
```

SQL

DDL DML DRL TCL DCL

DDL - Data Definition Language -- Auto Commit

create

alter

rename

truncate

drop

DML - Data Manipulation Language -- User commit

Insert

update

delete

Merge

DRL - Data Retrival language:

select

TCL - Transaction Control Language

commit

rollback

savepoint

DCL - Data Control Language (DBA)

Grant

Revoke

```
create table table_name
(
column_1 data_type,
column_2 data_type,
column_3 data_type,
.
.
.
column_n data_type
);
```

```
create table customer
(
cust_id number(6),
cust_name varchar2(30),
mobile_no  number(10),
dob date,
city varchar2(100),
email_id varchar2(100)
);
```

```
insert into table_name
(column1,column2,column3)
values
(value1,value2,value3);
```

```
create table customer
(
cust_id number(6),
cust_name varchar2(30),
dob date,
mobile number(10),
address varchar2(100)
);
```

```
select * from customer;
```

```
insert into customer
(CUST_ID,cust_name,dob,mobile,address)
values
(100000,'Arun',to_date('09/12/1992','mm/dd/yyyy'),9090909090,'Chennai');
```

```
select * from customer;
```

```
rollback;

commit;

insert into customer
values
(100001,'Kannan',to_date('09/11/2000','mm/dd/yyyy'),8132437493,'Chennai');

insert into customer
values
(100002,'Radha',to_date('09/24/2012','mm/dd/yyyy'),1348374989);
--SQL Error: ORA-00947: not enough values

insert into customer
(CUST_ID,cust_name,dob,mobile)
values
(100002,'Radha',to_date('09/24/2012','mm/dd/yyyy'),1348374989);

commit;

update table_name
set column_name=value
where condition;
```

```
update customer
set address='Hydrabad';


rollback;


select * from customer;

update customer
set address='Hydrabad'
where cust_id=100002;


commit;


-- Add a column


alter table table_name
add column_name data_type;




alter table customer
add zip number(6);
```

```
select * from customer;

--drop a column

alter table table_name
drop column column_name;


alter table customer
drop column address;



-- Rename a table

rename old_table_name to new_table_name;

rename customer to customer_details;

select * from customer_details;



-- rename a column

alter table table_name
rename column old_name to new_name;
```

alter table customer_details

rename column mobile to mobile_no;

---

1. bkp a table

2. truncate base table

3. modify data type

4. Restore the data

5. drop bkp table
